BAB II

LANDASAN TEORI

2.1 Game

Dalam bagian ini akan dibahas mengenai definisi Game, klasifikasi Game berdasarkan *genre*-nya yang terdiri dari *Role Playing Game, First Person Shooting, Third Person Shooter, Strategy, Simulation, Tycon, Racing, Action Adventure, Arcade, Fighting Game, Sport.*

2.1.1 Pengertian Game

Dalam kamus bahasa Indonesia "Game" adalah permainan. Permainan merupakan bagian dari bermain dan bermain juga bagian dari permainan keduanya saling berhubungan. Permainan adalah kegiatan yang kompleks yang didalamnya terdapat peraturan, *play* dan budaya. Sebuah permainan adalah sebuah sistem dimana pemain terlibat dalam konflik buatan, disini pemain berinteraksi dengan sistem dan konflik dalam permainan merupakan rekayasa atau buatan, dalam permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain dan menentukan permainan. Game bertujuan untuk menghibur, biasanya game banyak disukai oleh anak – anak hingga orang dewasa. Games sebenarnya penting untuk perkembangan otak, untuk meningkatkan konsentrasi dan melatih untuk memecahkan masalah dengan tepat dan cepat karena dalam game terdapat berbagai konflik atau masalah yang menuntut kita untuk menyelesaikannya dengan cepat dan tepat. Tetapi game juga bisa merugikan karena apabila kita sudah kecanduan game kita akan lupa waktu dan akan mengganggu kegiatan atau aktifitas yang sedang kita lakukan [5].

2.1.2 Klasifikasi Game

Klasifikasi Game berdasarkan *genre*-nya terbagi menjadi 10 jenis, yaitu *Board Game, Arcade, Action, Strategy, Shooting, Fighting, Racing, Simulation, Real Time Strategy, Role Playing Game, Masive Multiplayer Online* [6].

1. Board Game

   Board game atau dapat kita istilahkan sebagai "permainan papan". Pada jenis *game* ini, pemain diberikan sebuah tampilan yang berisi tentang masalah untuk diselesaikan. Contoh dari *game* ini antara lain : *Pipe dream*, Catur, permainan kartu dan sebagainya.

2. Arcade

   Game bertipe *arcade* merupakan *game* yang menguji kecepatan tangan pemainnya. Pada permainan bertipe *arcade*, semakin tinggi level permainan, permainan akan berjalan semakin cepat dan sulit. Contoh dari *game* bertipe *arcade* adalah *zuma*, *pacman*, *Arcanoid* dan sebagainya.

3. Action

   Berbeda dengan *game* bertipe *arcade*, *game* bertipe *action* menjadikan pemain mengendalikan karakter utama dalam *game* tersebut melakukan beberapa kegiatan seperti melompat, menembak dan sebagainya. Contoh dari *game* bertipe *action* adalah *super mario*, petualangan *paddle pop* dan sebagainya.

4. Shooting

   Game bertipe *shooting* sebagian besar menggunakan *mouse* sebagai alat pengendalinya. Pada *game* ini pemain seolah-olah berperan sebagai penembak atau pemain mengendalikan seorang penembak. Contoh *game* bertipe *shooting* adalah *Duck Hunt*, *Counter Strike* dan sebagainya.

5. Fighting

   Game bertipe *fighting* pada dasarnya sama dengan *game* bertipe *action*, hanya saja *game* bertipe *fighting* pemain mengendalikan sebuah karakter untuk berkelahi dengan karakter lain sampai salah satu karakter kalah. Contoh *game* bertipe *fighting* adalah *street fighter*.

6. Racing

   Game bertipe *racing* pada dasarnya adalah sebuah permainan menggerakkan kamera. Pemain diberikan sebuah kendaraan atau sejenisnya untu menempuh rute tertentu. Contoh dari *game* bertipe *racing* adalah *Need For Speed*.

7. Simulation

   Game bertipe simulasi adalah sebuah *game* yang mensimulasikan suatu kegiatan yang sesungguhnya. Contoh dari *game* bertipe simulasi adalah *tycoon*, simulator pesawat, *burger empire* dan sebagainya.

8. Real Time Strategy

   Game bertipe RTS memposisikan pemain sebagai seorang pemimpin yang mengatur sesuatu (bisa berupa pasukan, koloni, kerajaan dan sebagainya). Contoh dari *game* bertipe RTS antara lain *war commander*, *empire earth* dan sebagainya.

9. Role Playing Game

   Pada *game* bertipe RPG pemain memerankan sebuah karakter dalam *game*. Berbeda dengan *game* bertipe *action*, pada game RPG hal yang diutamakan adalah cerita dalam *game*. Selain itu di dalam *game* bertipe RPG biasanya terdapat *subgame* dengan tipe lain. Contoh *game* RPG adalah *Zelda*, *Megaman* dan sebagainya.

10. MMO

    MMO (Masive Multiplayer Online) merupakan *game* yang dapat dimainkan secara bersama-sama pada internet *browser*. Hal yang diutamakan dalam sebuah *game* bertipe MMO adalah kebersamaan dengan pemain lain. Contoh *game* bertipe MMO adalah *Ragnarok*.

2.2 Wayang

Dalam bagian ini akan dibahas mengenai penjelasan tentang kesenian wayang yang menjadi kebudayaan tradisional di Indonesia dan sejarah Gatotkaca dari mulai beliau lahir hingga wafat di medan perang.

2.2.1 Penjelasan Kesenian Wayang

Asal-usul dan perkembangan wayang tidak tercatat secara akurat seperti sejarah. Namun orang selalu ingat dan merasakan kehadiran wayang dalam kehidupan masyarakat. Wayang akrab dengan masyarakat sejak dahulu hingga sekarang, karena memang wayang merupakan salah satu buah usaha akal budu bangsa Indonesia. Wayang tampil sebagai seni budaya tradisional, dan merupakan puncak budaya daerah.

Periodisasi perkembangan budaya wayang bermula ketika nenek moyang bangsa Indonesia masih menganut animisme dan dinamisme. Dalam alam kepercayaan animisme dan dinamisme ini diyakini roh orang yang sudah meninggal masih tetap hidup, dan semua benda itu bernyawa serta memiliki kekuatan roh-roh itu bisa bersemayam di kayu-kayu besar, batu, sungai, gunung dan lain-lain.

Kekuatan utama budaya wayang, yang juga merupakan jati dirinya adalah kandungan falsafahnya. Wayang yang tumbuh dan berkembang sejak lama itu ternyata berhasil menyerap berbagai nilai-nilai keutamaan hidup dan terus dapat dilestarikan dalam berbagai pertunjukan wayang. Proses akulturasi kandungan isi wayang itu meneguhkan posisi wayang sebagai salah satu sumber etika dan falsafah yang secara tekun dan berlanjut disampaikan kepada masyarakat. Oleh karena itu, ada pendapat wayang itu tak ubahnya sebagai buku falsafah, yaitu falsafah Nusantara yang bisa dipakai sumber etika dalam kehidupan pribadi dan bermasyarakat [7].

2.2.2 Sejarah Gatotkaca

Gatotkaca adalah putra dari Bima dan Arimbi. Karena menurun dari wujud ibunya, maka separuh badannya merupakan raksasa yang mana hal ini banyak memberi kesaktian dan membuat dirinya menjadi seorang ksatria penting di medan perang. Pada masa kecil Gatotkaca, yang bernama Bambang Tetuka dijadikan jago para Dewa untuk menghadapi penyerang nirwana. Gatotkaca lalu dimasukkan oleh batara Narada ke kawah Candradimuka bersama dengan berbagai pusaka baja kahyangan, sehingga saat keluar dari kawah, Gatotkaca yang tadinya berwujud raksasa berubah menjadi ksatria yang gagah.

Gatotkaca lalu menjadi raja menggantikan ibunya di negara Pringgondani. Gatotkaca menikah dengan Dewi Pergiwa dan mempunyai putra bernama Sasikirana. Dari istri yang lain, Ahilawati, Gatotkaca memiliki anak bernama Barbarika. Gatotkaca memiliki beberapa ajian yaitu Aji Narantaka, Aji Brajadenta, Aji Brajamusti. Selain itu, dia menjadi penanggung jawab keamanan udara di kerajaan amarta, karena kemampuannya yang bisa terbang.

Pada saat perang Baratayudha, Adipati Karna berangkat perang pada waktu malam hari, dan hanya Gatotkaca yang dianggap bisa menandinginya karena dada Gatotkaca bisa bersinar akibat dari baju yang dipakainya. Pada perang inilah akhirnya Gatotkaca tewas oleh Adipati Karna yang memiliki pusaka ampuh bernama Kuntawijayadanu. Sebelum mati Gatotkaca mempunyai permintaan, yaitu bersedia mati tetapi harus diganti dengan kematian dari 1000 prajurit musuh. Tubuh Gatotkaca lalu jatuh dari angkasa kemudian tepat mengenai kereta Adipati Karna. Sebagai akibatnya, pecahan kereta ini mencederai 1000 prajurit Astina sehingga semuanya tewas seketika [8].

2.3 Brute-Force

Algoritma *brute force* merupakan salah satu algoritma pemecahan masalah dengan strategi solusi langsung (direct solution strategies). Dengan kata lain algoritma *brute force* memecahkan masalah dengan sangat sederhana, langsung dan dengan cara yang jelas (obvious way). *Brute force* adalah sebuah pendekatan yang lempang (straightforward) untuk memecahkan suatu masalah, biasanya didasarkan pada pernyataan masalah (problem statement) dan definisi konsep yang dilibatkan [9].

Kelemahan Algoritma *Brute Force* yaitu lemah dalam sisi penggunaan memori pada komputer jika data yang diinginkan berada dalam titik tengah ke atas (> n/2). Yang dimaksud dengan titik tengah ke atas yaitu titik dimana ketika data yang akan dicari berjumlah 10.000 dan *brute force* melakukan perulangan pencarian data lebih dari 10.000/2 = 5.000 kali. *Brute Force* menjadi sangat efektif ketika data yang akan dicari berada pada titik tengah ke bawah (< n/2).

Implementasi Algoritma *Brute Force* dalam pencocokan pola *string* sederhana dengan memeriksa dan membandingkan *Pattern* yang dicari dengan *Pattern* yang ada di *String*. Bila hasil cocok, maka program melakukan keluaran sesuai perintah, sedangkan bila hasil belum ditemukan maka program akan melakukan perulangan hingga hasil ditemukan [10].

Contoh dalam bahasa C :

```
#include <stdio.h>
int main()
{
  //String
  char kata[11];
  kata = "ihaveadream";

  //Pola yang dicari
  char cari[3];
  cari = "ead";
  return 0;
}
```

```
  int i;
  int indeks;

  for(i=0;i<kataLength-cariLength+1;i++)
  {
    int j=0;
    while(j<cariLength && kata[i+j] == cari[j])
    {
      j++;
      if(j >= cariLength)
```

```
      {
        indeks = i;
        //posisi pattern ditemukan = i
      }
    }
  }
```

Maka cara kerja Algoritma *Brute Force* dijelaskan pada gambar berikut ini :



Gambar 2.1. Cara kerja Algoritma *Brute Force*

Penyelesaian pada algoritma *Brute-Force* dalam pembuatan game terbilang cukup mudah dipahami. Perbandingan posisi dan menentukan langkah selanjutnya [11]. Algoritma :

1. Cek apakah tujuan ada disebelah kanan atau tidak,
2. Jika ya, bergerak ke kanan,
3. Jika tidak, bergerak ke kiri,
4. Cek apakah tujuan ada disebelah atas atau tidak,
5. jika ya, bergerak ke atas,
6. Jika tidak bergerak ke bawah

Karakteristik dari Algoritma *Brute-Force* antara lain :

1. Algoritma *Brute-Force* umumnya tidak “cerdas” dan tidak mangkus, karena ia membutuhkan jumlah langkah yang besar dalam penyelesaiannya. Kadang-kadang algoritma *Brute-Force* disebut juga algoritma naif.
2. Algoritma *Brute-Force* seringkali merupakan pilihan yang kurang disukai karena ketidakmangkusannya itu, tetapi dengan mencari pola-pola yang mendasar, keteraturan, atau trik-trik khusus, biasanya akan membantu dalam menemukan algoritma yang lebih cerdas dan lebih mangkus.
3. Untuk masalah yang ukurannya kecil, kesederhanaan *Brute-Force* biasanya lebih diperhitungkan daripada ketidakmangkusannya
4. Algoritma *Brute-Force* seringkali lebih mudah diimplementasikan daripada algoritma yang lebih canggih, dan karena kesederhanaannya, kadang-kadang algoritma *Brute-Force* dapat lebih mangkus.

2.4 Tools

Pada sub bab ini akan dijelaskan tools yang digunakan dalam Pembangunan Game 2D Gatotkaca Evolution ini antara lain penjelasan mengenai OOP, UML, Java, Java SDK dan Eclipse.

2.4.1 Objek Oriented Programing

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek dan pengujian berorientasi objek [12].

Keuntungan menggunakan metodologi berorientasi objek adalah :

1. Meningkatkan produktivitas karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut.
2. Kecepatan pengembangan karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengkodean.
3. Kemudahan pemeliharaan karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.
4. Adanya konsistensi karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodeaan.
5. Meningkatkan kualitas perangkat lunak karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.4.2 Unified Modeling Language (UML)

Unified Modeling Language (UML) adalah bahasa pemodelan standar. (Chonoles, 2003: bab 1) mengatakan sebagai bahasa, berarti UML memiliki sintaks dan semantik. UML bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. UML diaplikasikan antara lain untuk :

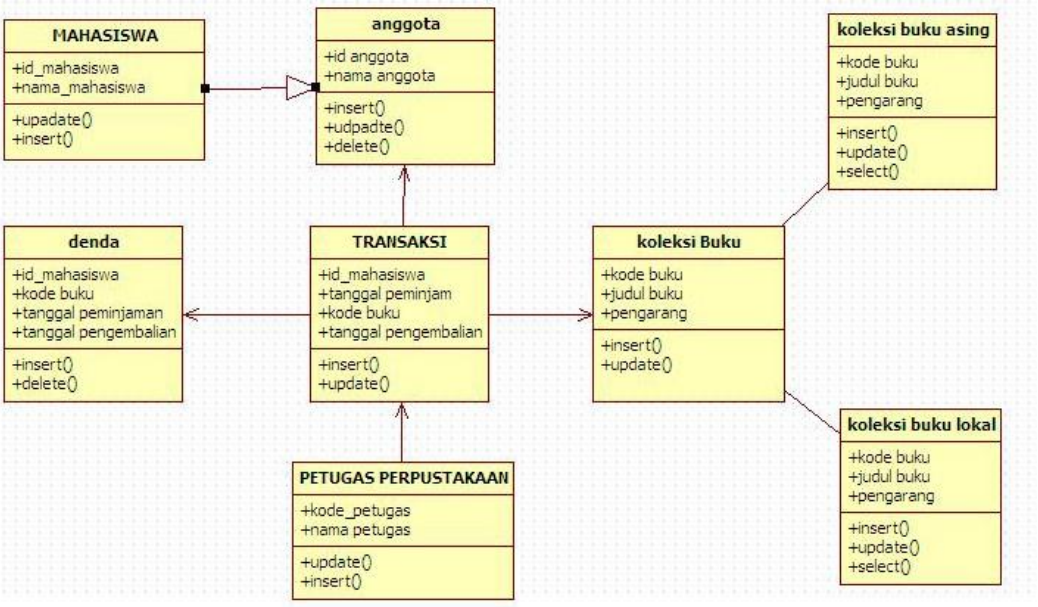
1. Merancang perangkat lunak
2. Sarana komunikasi antara perangkat lunak dengan proses bisnis
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem
4. Mendokumentasi sistem yang ada, proses-proses dan organisasinya.

Blok pembangunan utama UML adalah diagram. UML memungkinkan para anggota team untuk bekerja sama dengan bahasa model yang sama dalam

mengaplikasikan beragam sistem. Intinya, UML merupakan alat komunikasi yang konsisten dalam mensuport para pengembang sistem [13].

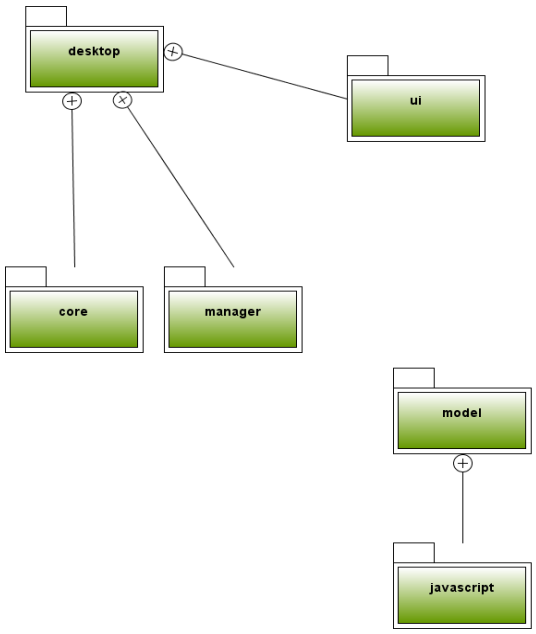
UML menyediakan sembilan jenis diagram, antara lain :

1. Diagram Kelas : diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi.



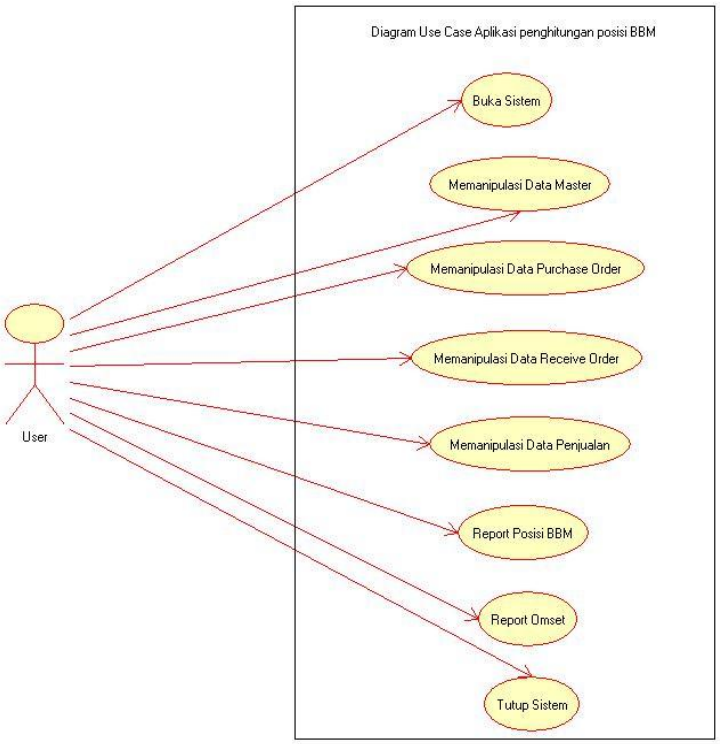
Gambar 2.2. Diagram Kelas

2. Diagram Paket : diagram ini memperlihatkan kumpulan kelas-kelas, merupakan bagian dari diagram komponen



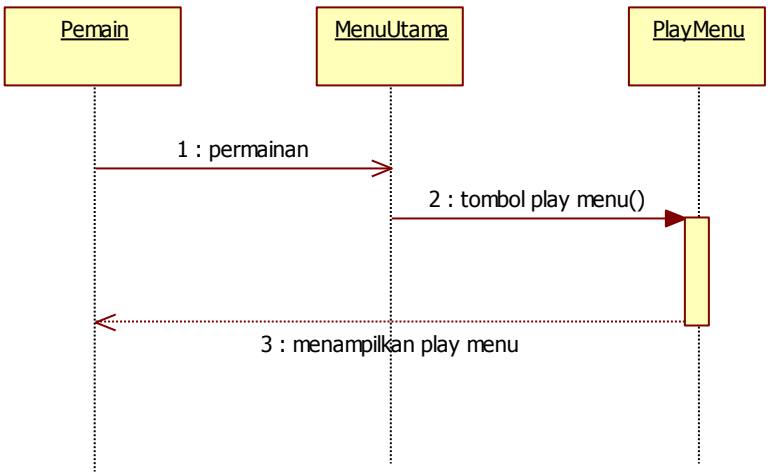
Gambar 2.3. Diagram Paket

3. Diagram *Use-Case* : diagram ini memperlihatkan himpunan *use-case* dan aktor-aktor.



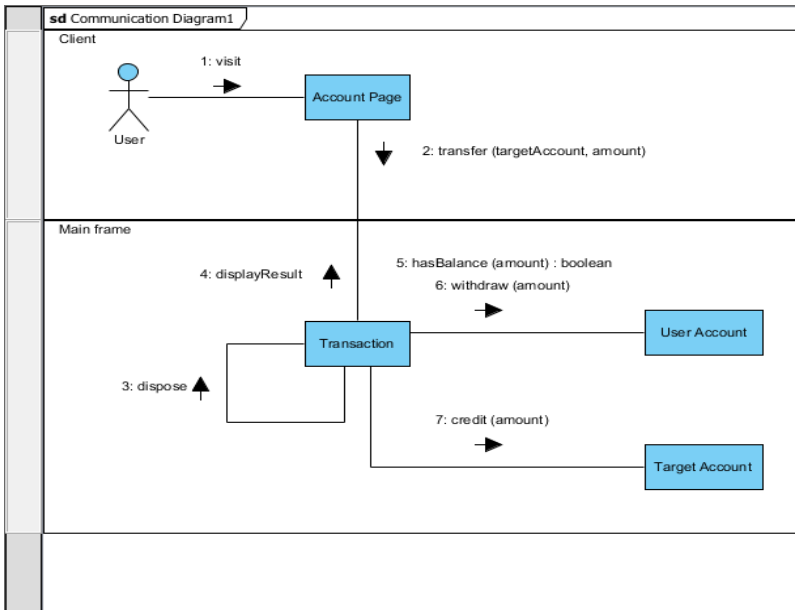
Gambar 2.4. Diagram *Use Case*

4. Diagram *Sequence* : diagram ini adalah diagram interaksi yang menekankan pada pengiriman pesan dalam suatu waktu tertentu.



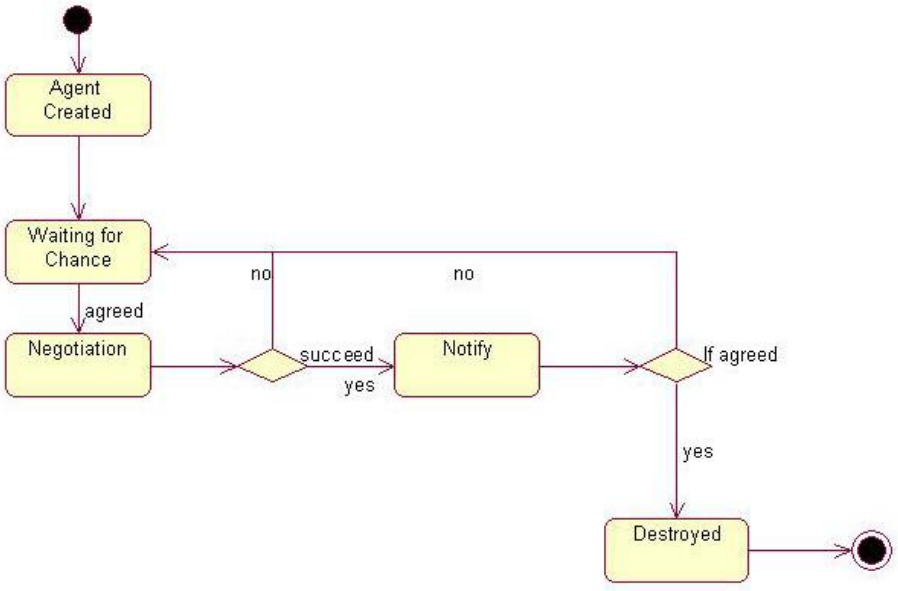
Gambar 2.5. Diagram *Sequence*

5. Diagram Komunikasi : diagram ini menekankan organisasi struktural dari objek-objek yang menerima serta mengirim pesan.



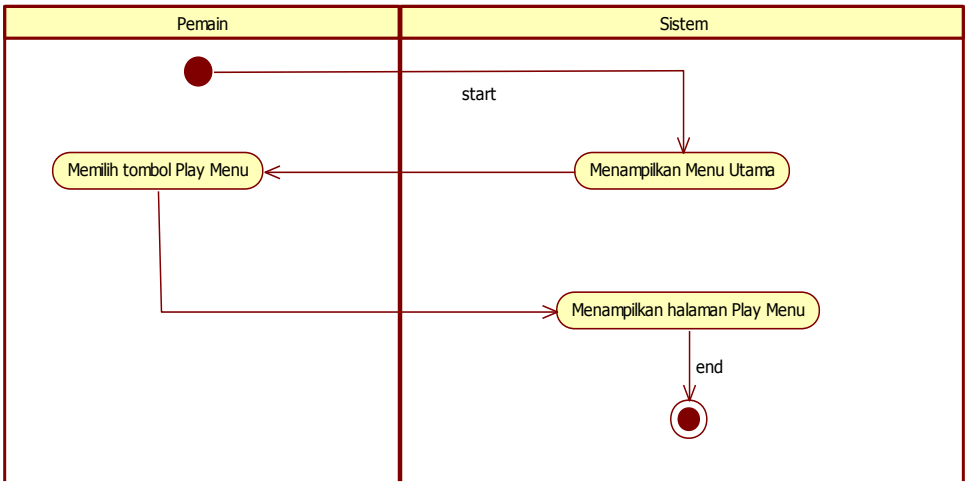
Gambar 2.6. Diagram Komunikasi

6. Diagram *Statechart* : diagram ini memperlihatkan keadaan-keadaan pada sistem, memuat status, transisi, kejadian serta aktivitas.



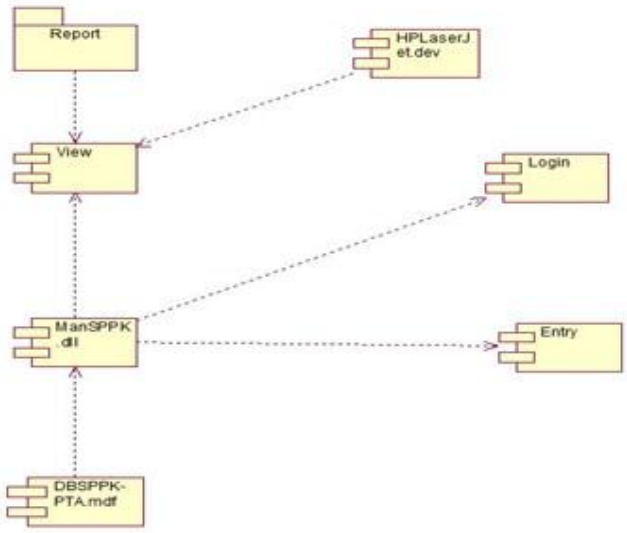
Gambar 2.7. Diagram *Statechart*

7. Diagram Aktivitas : diagram ini adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem.



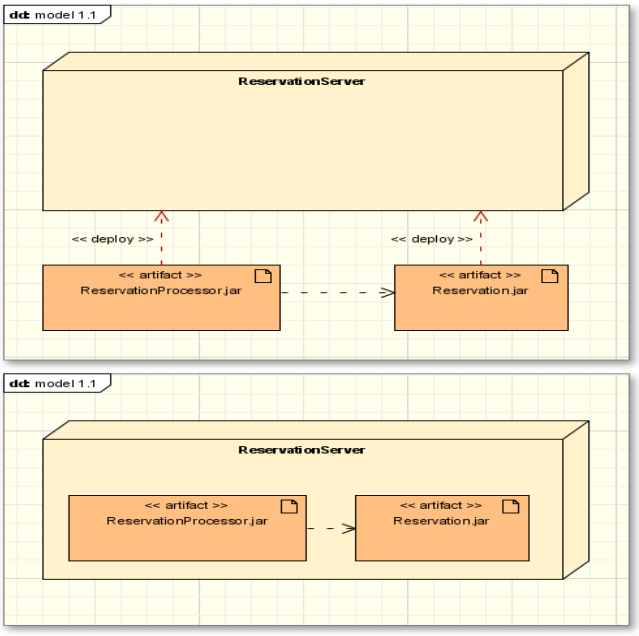
Gambar 2.8. Diagram Aktivitas

8. Diagram Komponen : diagram ini memperlihatkan organisasi serta kebergantungan sistem/perangkat lunak pada komponen-komponen yang telah ada sebelumnya.



Gambar 2.9. Diagram Komponen

9. Diagram *Deployment* : diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan.



Gambar 2.10. Diagram *Deployment*

2.4.3 Java

Java merupakan bahasa murni *object oriented*. Dalam Java struktur program berada di dalam class. Java memiliki syntax yang pada dasarnya sama dengan bahasa C/C++, namun Java lebih sederhana karena kemampuan *multiple inheritance, overload operator, preprocessor,* serta *pointer* telah dihilangkan di dalam bahasa Java. Java adalah bahasa *multi-platform* yang artinya dapat dijalankan oleh bermacam-macam *platform* dengan sistem operasi yang berbeda-beda. Bahasa Java juga mendukung *multithreading* yaitu program dapat dibuat untuk dijalankan oleh *thread* tertentu [14].

Pada dasarnya terdapat tiga garis besar kelompok program yang dapat dibuat menggunakan bahasa Java, yaitu :

1. *Applet* sering disebut sebagai aplikasi mini.
2. Aplikasi Java, dibagi menjadi :
    a. Aplikasi GUI (Graphical User Interface) yaitu aplikasi yang tampilannya memakai grafik. Aplikasi ini dapat dijalankan di komputer lokal.
    b. Aplikasi *Command-Line* yaitu aplikasi yang tampilannya sebatas teks.
3. *Package* atau disebut juga *library* Java. Package sendiri tidak untuk dijalankan, ia hanya menyediakan sekumpulan *class-class* Java yang dianggap berguna dan umum dipakai kembali.

2.4.4 Java 2 Software Development Kit (Java SDK)

Java 2 SDK adalah satu set perangkat lunak yang digunakan untuk manajemen dan membangun berbagai macam aplikasi Java [14]. Java 2 SDK mempunyai *file-file* berupa *tool* dasar, diantaranya :

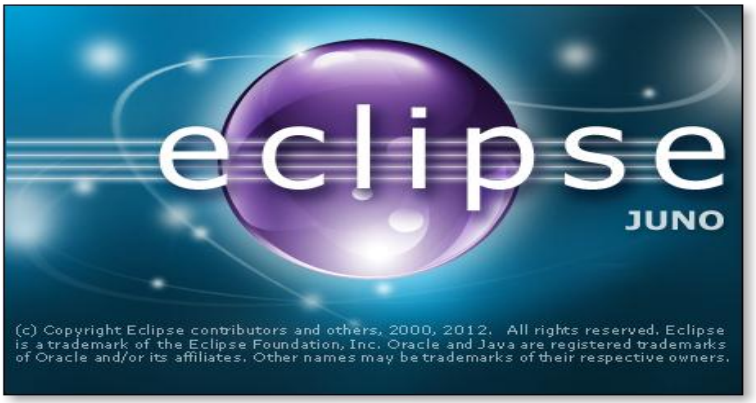
1. *Compiler* bernama javac.
2. *Launcher* atau *interpreter* untuk menjalankan aplikasi Java bernama java.
3. *Debugger* Java bernama jdb.
4. Penampil applet bernama appletviewer.
5. *Disassembler* untuk file .class bernama javap.

6. *Generator* untuk dokumentasi API bernama javadoc.
7. Manajer file-file JAR (Java Archive) bernama jar.
8. *Utility* untuk mendeteksi konflik Jar bernama extcheck.
9. *Generator file header* dan *stub* bernama javah.

2.4.5 Eclipse IDE

Eclipse merupakan komunitas *open source* yang bertujuan menghasilkan *platform* pemrograman terbuka. *Eclipse* terdiri dari *framework* yang dapat dikembangkan lebih lanjut, peralatan bantu untuk membuat dan me*manage software* sejak awal hingga diluncurkan. *Platform Eclipse* didukung oleh ekosistem besar yang terdiri dari *vendor* teknologi, *start-up* inovatif, universitas, riset institusi serta individu. Banyak orang mengenal *Eclipse* sebagai IDE (integrated development environment) untuk bahasa Java, tapi *Eclipse* lebih dari sekedar IDE untuk Java [15]. Komunitas *Eclipse* memiliki lebih dari 60 proyek *open source*. Proyek-proyek ini secara konsep terbagi menjadi 7 categori :

1. Enterprise Development
2. Embedded and Device Development
3. Rich Client Platform
4. Rich Internet Applications
5. Application Frameworks
6. Application Lifecycle Management (ALM)
7. Service Oriented Architecture (SOA)



Gambar 2.11. Tampilan Eclipse IDE

2.4.6 Slick2D

Slick2D adalah *multi-platform library* untuk pengembangan *game* dua dimensi yang berada di bawah naungan LWJGL(Light-Weight Java Game Library). Slick2D menyederhanakan proses pengembangan game seperti *loop*, *rendering*, *updating*, *frame setup* dan pembuatan basis permainan. Slick2D juga menawarkan beberapa fitur yang LWJGL tidak berikan, seperti pembuatan partikel dan integrasi dengan *Tiled*. Pengembang dari semua tingkatan dapat menggunakan *library* Slick2D, karena kesederhanaan yang tidak dapat ditemukan disebagian besar *library* lainnya, sehingga seseorang yang tidak memiliki pengetahuan teknis dalam membuat *game* pun terbantu dengan *library* ini jika ingin belajar membuat *game* [16].

2.4.7 Adobe Photoshop

Adobe Photoshop adalah *software* pengolah foto yang sangat populer dan telah banyak para pengguna program ini yang mengakui kehebatan dari Adobe Photoshop. Dengan Photoshop kita dapat memamnipulasi sebuah foto agar terlihat lebih menarik dan mengesankan, dengan fasilitas-fasilitas yang ada pada Photoshop, kita juga dapat menggabung foto serta memberikan efek-efek khusus untuk mempercantik tampilan foto.

Tampilan lembar kerja Photoshop terdiri dari *Menu Bar*, *Panel Control* dan *Panel Tools*. *Menu Bar* adalah baris paling atas yang berisi perintah yang disediakan oleh sebuah program yang mendukung kinerja dari program tersebut. *Panel Control* adalah bagian yang berisi sekumpulan tombol atau pilihan yang letaknya berada tepat di bawah *Menu Bar* yang digunakan untuk mengatur sifat dan kinerja dari piranti yang terpilih di *Panel Tools*. *Panel Tools* adalah sebuah kumpulan piranti yang digunakan untuk mengolah gambar, seperti melakukan seleksi area, memotong bidang gambar, maupun menghapus bidang gambar [17].